Algorithmic Issues in Modeling Motion

Pankaj K. Agarwal, Leonidas J. Guibas,
Herbert Edelsbrunner, Jeff Erickson, Michael Isard, Sariel Har-Peled,
John Hershberger, Christian Jensen, Lydia Kavraki, Patrice Koehl, Ming Lin,
Dinesh Manocha, Dimitris Metaxas, Brian Mirtich,
David Mount, S. Muthukrishnan, Dinesh Pai, Elisha Sacks,
Jack Snoeyink, Subhash Suri, Ouri Wolfson *

Abstract

This report presents the results of the workshop on Algorithmic Issues in Modeling Motion, funded by NSF and ARO, held on August 6 and 7, 2000 at Duke University, Durham, NC. The report identifies research areas in which motion plays a pivotal role, summarizes the challenges that lie ahead in dealing with motion, and makes a number of specific recommendations to address some of the challenges presented.

*Pankaj K. Agarwal, Duke University, pankaj@cs.duke.edu; Leonidas J. Guibas, Stanford University, guibas@cs.stanford.edu; Herbert Edelsbrunner, Duke University, edels@cs.duke.edu; Jeff Erickson, University of Illinois, Urbana-Champaign, jeffe@cs.uiuc.edu; Michael Isard, Compaq Research Labs, misard@robots.ox.ac.uk; Sariel Har-Peled, University of Illinois, Urbana-Champaign, sariel@cs.uiuc.edu; John Hershberger, Mentor Graphics john_hershberger@mentorg.com; Christian Jensen, University of Aalborg csj@borg.cs.auc.dk; Lydia Kavraki, Rice University, kavraki@cs.rice.edu; Ming Lin, University of North Carolina, Chapel Hill, ming@cs.unc.edu; Dinesh Manocha, University of North Carolina, Chapel Hill, dm@cs.unc.edu; Dimitris Metaxas, University of Pennsylvania dnm@central.cis.upenn.edu; Brian Mirtich, MERL mirtich@merl.com; David Mount, University of Maryland, mount@cs.umd.edu; S. Muthukrishnan, AT&T, muthu@research.att.com; Dinesh Pai, University of British Columbia, pai@cs.ubc.ca; Elisha Sacks, Purdue University, eps@cs.purdue.edu; Jack Snoeyink, University of North Carolina, Chapel Hill snoeyink@cs.unc.edu; Subhash Suri, University of California, Santa Barnara, suri@cs.ucsb.edu; Ouri Wolfson, University of Chicago wolfson@eecs.uic.edu.

1 Introduction and Background

Motion is ubiquitous in the physical world. Computational models of physical objects and processes, from protein folding in biology, to assembly planning in manufacturing, from connectivity in mobile communications networks to location based services in spatial databases, from ecological models in environment science to galactic evolution in astrophysics, must deal with representing mobile data and their evolution over time. It is instructive to draw a comparison with another modality that permeates modeling the physical world as much as motion, namely shape. Over the years computational shape representations have been intensely studied and a variety of approaches were developed for modeling the different shapes that arise in applications. Entire disciplines, such as CAGD (computer-aided geometric design) evolved, in this case devoted to the of study of the smooth shapes needed by the automotive and aerospace industries. What is striking when we compare motion to shape, however, is how rich and complex the space of representations for motion can be. For one, motion is often described intensionally — by specifying what the motion needs to accomplish, and not what the motion is. For another, motion happens over time and in the physical world unpredictable events can happen that change the evolution of a system — this on-line character of motion must be modeled as well.

Both military and civilian applications (digitized battlefields, automatic target recognition, mobile communications, virtual environments, animation, physics-based simulation, animation of deformable objects — to name a few) have raised a wide range of algorithmic issues in modeling motion and coping with continuously moving objects. Proper algorithmic techniques are lacking for dealing with many of these issues. Despite a flurry of activity in several areas in the last few years on modeling motion and on developing efficient algorithms that deal with moving objects, a unified algorithmic theory of motion that permeates across multiple disciplines and applications has been missing. A workshop, sponsored by NSF and ARO, held on August 6 and 7, 2000 at Duke University was aimed at bringing together a set of researchers to survey the state of the art in motion modeling and to propose initiatives that would create a firm algorithmic theory for the processes of acquiring, modeling, reasoning about, planning, manipulating, and executing motion.

2 Goals

Geometric computing lies at the core of many areas highlighted in the IT^2 initiative: information visualization, advanced science and engineering computation, and computational and algorithmic methods. Efficient algorithms and data structures for handling geometric data are needed in almost every computational field that deals with the physical world, including computer graphics, computer vision, robotics, geographic information systems, spatial databases, molecular biology, and scientific computing — not to mention entertainment and computer games.

Although a vast amount of work has been done on modeling, analyzing, searching, processing, and visualizing geometric objects, most of the work has by and large focussed on handling static objects and on dynamic data structures, which can update the information as objects are inserted or deleted at discrete times. These approaches are, however, not suitable for handling moving or deforming objects because either the algorithm has to be executed continuously, or the solution returned will be at times obsolete. The on-line nature of motion makes it difficult to consider time simply as another dimension and regard the moving object as a static object in space-time. Algorithms that can handle continuous motion, what we call *kinetic algorithms*, are needed in such applications.

Most of the work to date on modeling motion and kinetic algorithms has been scattered among a number of fields. This report argues that a synergy among these areas is needed, since many issues are common to all the areas; furthermore many real-world applications require the integration of techniques from a wide range of disciplines. This report includes sections devoted specifically to computational geometry, mesh generation, physical simulation, biology, computer graphics, computer vision, robotics, spatio-temporal data-bases, and mobile wireless networks. It concludes by highlighting a number of challenges common across several of these fields and making a set of recommendations to further advance the integration across disciplines and push the state of the art in motion algorithms.

3 Areas and Problems

We have identified nine main areas in which motion plays a pivotal role. Although there are several common issues that arise in many of these areas (for instance, motion representation, dealing with uncertainty, etc.), some of the issues are particular to specific domains. This section presents the state-of-the-art concerning motion modeling in each of these areas and describes some of the main open problems in each of them.

3.1 Computational Geometry

Earlier work on moving objects in computational geometry, initiated by a paper by Atallah in 1985 [7], focused on bounding the number of combinatorial changes in geometric structures as the objects move along prescribed trajectories [6]. In the late 1980s algorithms were developed for computing the changes in geometric structures as objects undergo motion. However, these results assume an off-line model of motion in which the trajectories are given in advance, which is a relatively uncommon situation. In practice, researchers were using fixed-rate sampling methods to handle moving objects, but without a principled to way to select the proper time step. The kinetic data structure (KDS) framework introduced by Guibas *et al.* [9] alleviated many of the problems with the off-line and fixed-rate-sampling methods. The main idea in the kinetic framework is that even though objects move continuously, the relevant *combinatorial* structure changes only at certain discrete times and therefore one does not have to update the data structure continuously. The validity of the combinatorial structure is certified by a number of elementary assertions about the state of the system called *certificates*. *Kinetic updates* are performed on the data structure only when certain *kinetic events* occur, corresponding to certificate failures. In contrast to fixed-time-step methods, in which the structure is updated at fixed time intervals and therefore the fastest moving object gates the time step for the entire system, a kinetic data structure is based on events determined by localized conditions that can be monitored at different rates.

Kinetic data structures have led to efficient algorithms and data structures for a wide range of problems. However, in order for KDS and other motion models to be made more realistic, several issues need to be addressed.

Motion sensitivity. The motions of objects in the world are often highly correlated and it behooves us to find representations and data structures that exploit such motion coherence. It is also important to find mathematical measures that capture how coherent motions are and then use this coherence as a parameter to quantify the performance of motion algorithms. If we do not do this, our algorithm design may be aimed at unrealistic worst-case behavior, without capturing solutions that exploit the special structure of the motion data that actually arise in practice. Most kinetic algorithms to date have been analyzed under the model where each object follows an independent smooth trajectory defined by a small set of parameters, and the algorithm's efficiency is measured by the worst-case behavior over all such possible motions. A challenging issue is to develop a class of kinetic *motion-*

sensitive algorithms, whose performance can be expressed as a function of how coherent the motions of the underlying objects are.

Trade-offs between complexity measures. The performance of a kinetic algorithm can be measured by the number of events it processes, the time spent in updating the structure at each event, and the size of the structure. Although optimal or near-optimal algorithms under these criteria have been proposed for a number of kinetic problems, including convex hull of a point set in the plane and closest pair, no optimal algorithms are known for a wide range of other problems e.g., convex hulls in $\mathbb{R}^3$, triangulation of points in $\mathbb{R}^2$, range searching, etc. In such cases, one would like to obtain a trade-off between different complexity measures. A natural trade-off would be between the compactness and the efficiency of a KDS. This is similar to the trade-off between the size and the query time of a static data structure — a query can be answered more efficiently by allowing more space. A similar issue is to improve the efficiency of a kinetic algorithm by maintaining a geometric structure implicitly. For example, we know of techniques for maintaining the triangulation of a point set in the plane implicitly by structures that process only quadratic number of events, but no such bounds are known if is required to maintain a triangulation explicitly.

Another possible trade-off is between efficiency and accuracy. How much efficiency can be gained by maintaining a geometric structure *approximately*? For example, it is known that the diameter of a set S of points (the maximum distance between a pair of points in S) moving in the plane can change quadratically many times [3], but recently it was shown [5] that if we maintain the diameter approximately, the number of events only depends on the approximation factor (and is independent of the number of points). Although some preliminary work has been done on trading off efficiency with accuracy, a more general theory is needed for approximating motion.

Flexible scheduling. The current KDS implementations maintain a global priority queue that stores the future events at which the structure may need to be updated. The events are processed in the sorted temporal order. The correctness of these methods relies on the assumption that all the events have been processed in this order. Under algebraic motion an event is typically a root of a polynomial of fixed degree, and thus we have to compute the roots of a polynomial. If we use floating-point arithmetic or numerical methods to estimate the roots, we compute them only approximately and cannot ensure that the events have been processed in the sorted order. On the other hand, using exact arithmetic and algebraic methods (such as Sturm sequences) for computing or isolating roots is quite expensive. Another source of difficulty in processing events in the sorted order is degeneracies in the input — a degeneracy causes multiple events to occur simultaneously. Little attention has been paid so far to these problems.

In many physical simulations the motion law of the objects is specified by an ordinary or partial differential equation. In such contexts polynomial trajectories can be at best short-term approximations of the actual trajectories, useful for conservative estimates of certificate failure times. The correctness of the approximation needs to be certified and tracked as well.

It is desirable to develop a framework where event-based scheduling can be intermixed with fixed time-stepping. The latter can be used whenever explicit motion prediction, apart from full integration of the equations of motion, is difficult. The current KDS repair mechanism strongly

depends on the assumption that it is invoked to repair *a single* certificate failure. In a time-step based scenario multiple certificates may fail; we need techniques that can repair a structure after multiple certificate failures.

Canonical vs. non-canonical structures. The complexity measures mentioned earlier are more suitable for maintaining *canonical* geometric structures, those that are uniquely defined by the position of the data, e.g., convex hull, closest pair, and Delaunay triangulation. In these cases the notion of external events (those that change the combinatorial structure of the attribute of interest) is well defined and is independent of the algorithm used to maintain the structure. On the other hand, suppose we want to maintain a triangulation of a moving point set. Since the triangulation of a point set is not unique, the external events depend on the triangulation being maintained, and thus depend on the algorithm. This makes it hard to analyze the efficiency of a kinetic triangulation algorithm. Most of the current approaches for maintaining non-canonical structures artificially impose canonicality and maintain the resulting canonical structure. But this typically increases the number of events. Another drawback of imposing a canonicality condition is that in it might make the structure global, which in turn would make the task of designing a distributed algorithm considerably harder.

In general, current techniques are too weak to analyze the performance of non-canonical structures and better techniques are needed. In [1] we made some progress in this direction, but these techniques are very problem specific. A better model for analyzing the lower bounds is needed. Some of the analysis techniques developed for analyzing the on-line algorithms could prove useful.

Decentralization. When we want to simulate motion over a large scale, e.g., the flow patterns of cars in a city or a chemical reaction among several macromolecules, the motion computation may have to be distributed over several processors/machines. Furthermore, if we are confirming certificates from sensor data, the sensors themselves may be geographically distributed over a large area and may be also constrained by communication bandwidth and power requirements. In such settings the normal centralized way of implementing KDSs cannot be used, as the communication, processing, and power capabilities of the central node will severely limit the scalability of the system. Most of the existing approaches assume the existence of a central event queue that keeps track of all the changes. In a distributed environment, reasoning about the state of the world and tracking the attribute of interest must take place in a distributed manner. Each processing node will be responsible for only some of the mobile data and will communicate information with other nodes as necessary. How to develop such algorithms even for simple problems remains a challenging issue, despite a few recent attempts in this direction.

The number of combinatorial changes. Although optimal or near-optimal bounds are known on the number of changes in the basic geometric structures such as convex hull, closest pair, diameter, and width of a point set, such bounds have remained elusive on many other structures, including Delaunay triangulation, minimum spanning tree, and alpha shapes. Moreover, most of the work has focused on bounding the number of changes in the worst-case, which rarely occurs in practice. There has been some work on bounding the number of changes when the trajectory of each point is randomly chosen from a reasonable distribution. It would be useful to study the number of changes

under more realistic assumptions on motion, e.g., when the motion of the points is correlated, or to develop a model in which the bounds are proved as a function of the complexity of the motion of objects.

3.2 Mesh Generation

Many physical and engineering problems are modeled by defining a set of partial differential equations over a continuous domain. Since it is infeasible to perform computer simulations over a continuous space, the domain is discretized by decomposing its interior into a mesh of simple and well-shaped elements such as boxes or simplices, the differential equations are approximated over the discretized domain using finite difference, element, or volume techniques, and then solved numerically. The error of a solution is estimated, and if necessary, the mesh is refined and the numerical simulation repeated. Not all meshes are equally good, because the error of discretization depends on the geometric shape and size of the elements of the mesh and the computational complexity of the simulation depends on the number of elements in the mesh and their overall quality. This has led to vast literature on mesh generation; see [10] for a recent survey.

In many applications, ranging from weather modeling to seismic analysis, from protein folding to heart modeling, the domain evolves with time, which raises the problem of mesh generation over time-varying domains. Depending on the application, either the mesh is updated dynamically over time, or time is considered as another dimension and the mesh is generated over the space-time domain.

Dynamic meshes. To maintain the mesh we must trace changes of the shape and repair the mesh at places where it becomes inadequate. We see at least three types of changes:

(i) Changes of *shape*, which can usually be accommodated by moving mesh vertices (and thus implicitly cells).

(ii) Changes of *curvature*, which cause distortions of the homeomorphism between the mesh and the underlying domain. In the case of triangulations we can repair such distortions by locally inserting or deleting simplices.

(iii) changes of *topology*, which require a local reorganization of the mesh connectivity.

In all cases, we require good mechanisms to predict or detect when these changes happen, as we already discussed for kinetic certificates in Section 3.1

Algorithms that maintain meshes under all three types of changes are rare [14], and they are sensitive to how quickly changes can happen. This gives rise to a hierarchy of motions or deformations, ordered by difficulty in maintaining a mesh algorithmically. A deformation may be best understood by considering the trajectory of the domain through a higher-dimensional space-time. For example, a deforming 2-dimensional surface in $\mathbb{R}^3$ sweeps out a 3-dimensional manifold in $\mathbb{R}^4$. The speed of the deformation is then captured by the slope of the manifold in the direction of time.

Meshing space-time domains. A possible alternative to dynamic meshing is meshing directly in space-time, which trades the dynamic nature of the problem for an extra dimension. *Space-time meshing* algorithms are becoming more desirable now as numerical algorithms for space-time problems are being developed. Recently, space-time discontinuous Galerkin methods have been proposed [30]. These methods rely on the *domain of influence* for dynamic data. The domain of influence for a point p in the space-time domain is a cone with apex p whose boundaries are determined by the maximum speed; the cone specifies the region in which the point p can lie in the future. In these methods, one needs to develop meshes that satisfy the so-called *cone constraint*, which requires the dihedral angle of each interior mesh face with respect to the space domain to be bounded by a specified parameter, which depends on the boundary of the cone. However, unlike many other techniques, the discontinuous Galerkin methods impose no restriction on the shape of elements of the mesh. Despite some recent work on one- and two-dimensional spatial domains, the problem of meshing over space-time domain remains largely open. Here are a few open problems:

- Many of the existing methods divide the time into fixed length intervals and construct the mesh within each layer. Can a more adaptive approach be developed?
- The current methods generate too many elements for the mesh; no technique with provable bound on the size of the mesh is known.
- In many material flow problems, the cone constraint is skewed or anisotropic. The current techniques assume uniform cone constraints. This brings the issue of modeling increasingly complex motion hierarchically, as in Section 3.1.
- Sometimes, e.g., in presence of shock waves, the cone constraint is discontinuous or it changes with time. In such cases, the mesh has to be adapted with time.

3.3 Physical Simulation

The applications of physical simulation can be divided into two broad categories. In one category are applications requiring physical simulation to reason about the real world for engineering, verification, and prediction. Here, accuracy is key, and many of the challenges involve the development of accurate models. Many of these, such as finite-element or boundary-element models, require the use of meshing techniques as discussed in Section 3.2. The second category of applications require physical simulation for producing 'plausible' motion; these applications include animation, games, content creation, and some kinds of education and training. Here, since the constraints on accuracy are more lax, modeling is less of a problem, and efforts have focused on other issues such as unstructured interaction and real-time speeds. Below are four challenges related to motion in physical simulation; the first three are relevant to both categories of applications, and the final one is relevant to the non-engineering applications.

One grand challenge of physical simulation is the combination of different state representations to solve complex, heterogeneous problems. Consider the different ways motion is represented through velocities. The velocities of all points on a rigid body are simultaneously described by the linear velocity of a single point on the body and an angular velocity. On the other hand, the velocity of a

deformable body is often represented by the linear velocity of a number of node points attached to the body. Finally, the velocity of fluid might be represented by linear velocities at lattice points fixed in space. How are these representations to be harmonized when studying, for example, the turbulence induced by a rigid body moving through a fluid, or the wrinkling of the clothes on an articulated human figure? Some progress has been made, but many problems remain in modeling interactions and constraints between objects with different underlying state and motion representations.

A second challenge is to design simulation systems than can effectively choose appropriate motion models and algorithms for the given situation. Perhaps a rigid body model is a reasonable approximation, or maybe a linear or even non-linear deformable model is needed. Inertial effects may or may not be negligible. There are a variety of ways to model friction with different properties and computational costs. Even after the model has been chosen, there might be alternative algorithms. For example, there are at least three families of methods for modeling rigid body contact (analytic/LCP formulations, penalty methods, and impulse-based methods), each with different strengths and weaknesses. By and large, today's simulation systems use a single model and a single algorithm for a given class of dynamic objects. Better if they could choose more judiciously among them on the fly. To do this, a system must know what are the important results of the simulation, what are the allowable tolerances, what are the sources of error in each algorithm, what are the bounds on these errors, and how do the errors propagate over time.

A third challenge is to parallelize simulation algorithms so that they can be scaled to bigger problems by adding more computational units. This is already possible for certain types of simulations involving finite element methods and computational fluid dynamics. It is harder for more unstructured systems such as those involving rigid bodies. Often, such systems can not be statically partitioned into independent subsystems, and furthermore the parts of the system that affect another part vary over time. A partitioning among CPUs needs to be a dynamic one. Better bounds on the motion of objects over time could make maintaining this partition easier. Also to be considered are load balancing issues: equally sized partitions are most useful.

A final challenge in physical simulation is to give better control to the user. Particularly when used for non-engineering applications, simulation is praised for producing complex motion while cursed for being an unwieldy, unpredictable tool. Despite living in a physical world, humans are poor at designing simulations to give them 'what they want;' in many cases the desired outcomes are not even physically possible. Despite a lot of work in this area, users still need better handles and hooks into the physical simulation process in order to steer its course. Often the user's desires are expressible in terms of motion constraints. Representations for these constraints are needed that are both intuitive to humans and which lead to tractable computer solutions. Also needed are algorithms that can let go of the true physics just enough to meet the constraints.

3.4 Biology

Molecules move and deform. Chemical processes essential to life critically depend on the ability of biomolecules to adopt different shapes over time. If we could realistically describe molecular motions, we would greatly improve our understanding of these temporal processes, including protein folding and molecular interactions (such as the assembly of protein complexes, protein-ligand

docking, etc.), representing some of the most important unsolved problems in biology. Simulating the motions of molecules poses its own challenges, beyond those covered in earlier sections.

Molecular dynamics simulations. At the atomic level, the physical principles underlying molecular motions are fairly well understood: ligands and proteins have primarily torsional degrees of freedom, and the forces on an individual atom can be expressed as sums of well-known potentials. Yet molecular dynamics simulations of a molecular system are still far too expensive computationally to allow us to follow a process over its complete time period. For example, a protein folds within a few seconds, while the longest published molecular dynamics simulation of a protein covers one microsecond. The reason that accurate molecular dynamics simulations remain a major challenge in computational biology is that they involve thousands of degrees of freedom in the molecule of interest, require many evaluations of potential energy functions that can be expensive to compute, and in addition need to take in account the solvent environment. In order to make progress, we must develop novel ways to represent deforming shapes and track their evolution over time, find techniques for describing the physics in terms of higher-level units than individual atoms, and use efficient approximations whenever justified to do so. More specifically, we need:

Implicit solvent potential energy functions. Molecular dynamics simulations that include a large number of water molecules around the solute remain the state of the art in this field. They are, however, inefficient, since a large fraction of the computing time is spent calculating a detailed trajectory of the solvent molecules, even though it is primarily the solute behavior that is of interest. Several semi-analytical implicit treatments of solvent have been proposed. They all rely on an energy term that is related to the solvent exposed surface area of each atom of the solute. Inclusion of such terms in a molecular dynamics simulation requires the calculation of accurate surface areas, as well as their analytical derivatives with respect to atomic position. Fast analytical methods are needed for these calculations [12].

Efficient updating procedures. Molecular dynamics simulations are based on solving the Newton equations of motion for all atoms of the system considered. These equations are solved numerically, using very short time steps of the order of a fraction of a femtosecond. Each step requires the evaluation of the total energy function of the system, as well as its derivatives with respect to atomic position. For standard energy terms such as Lennard-Jones and electrostatics interactions, only a fraction of all pairs of atoms need be considered. A crucial aspect for fast molecular dynamics simulation is to have access to fast algorithms for building and updating these interacting pairs of atoms.

Hierarchical representation of proteins. Very long simulation of molecular dynamics require approximations. One approach is to simplify the representation of the molecule of interest: simplified models of proteins have proved both popular and effective until the present time. A significant weakness of such models however is their inability to deal with topological features of the molecule in a meaningful way. We need new methods that build simplified models with geometrical persistence.

A motion-planning approach to protein folding. Molecular dynamics simulations follow molecular motions by solving a deterministic or stochastic system of equations. They provide very detailed analysis of short and usually small amplitude motions, with reasonable agreement with experimental data coming from structural biology. While it is expected that improvements of the computer technology and in the algorithms applied for solving these equations will increase the time span covered by these simulations, time scales on the order of milliseconds and seconds are still out of reach. As we already remarked, unfortunately these are the time scales of most interest for many important biological processes. Different approaches specific to these problems need to be explored.

There are large and ongoing research efforts whose goal is to determine the native folds of proteins (the protein structure prediction problem). Most approaches proceed in two steps. Firstly, a large collection of possible conformations (or decoys) for the protein of interest is built. Secondly, these conformations are screened using potential energy functions and the 'best' models define the predicted structures for the protein. In these procedures, the focus is set on thermodynamics (definition of low energy states) and not on the kinetics of the protein folding process (i.e., how the protein folds to its native conformation). A better understanding of the latter should not only facilitate protein structure prediction but also provide insights on how proteins function. Motion planning and more specifically probabilistic roadmap methods provide one approach to solving this problem. Application of such techniques to the protein folding problem requires:

Falsifiability studies. Motion planning usually starts by sampling the moving object's configuration space (C-space), and retaining those conformations that satisfy certain feasibility requirements. Roadmap nodes are then connected by finding the path of minimum energy between the starting point and the goal. Application of such a strategy to the protein folding problem is not intuitive and should be extensively tested. Interestingly, recent experimental results suggest that protein folding mechanisms and landscapes are largely determined by the topology of the native state and are relatively insensitive to details of the interatomic interactions. This dependence on low resolution structural features suggest that it should be possible to describe the physics of the folding process using relatively low resolution models. It remains that initial studies should focus on proteins for which kinetic data are available, as well as on proteins whose dynamics have been described.

Algorithmic developments. In the case of protein folding, the C-space is large, since the protein has a very large number of degrees of freedom. Fast algorithms for extensive sampling of the C-space of proteins must be developed. The protein folds based on geometric constraints (i.e., no self-collision) but also energetic constraints. Realistic energy functions are therefore needed, such as those used for molecular dynamics simulations — a point already mentioned. Algorithms are also needed to select roadmap nodes from possible configurations of the protein in its C-space. Such selection can be based on energetic and/or geometric criteria.

3.5 Computer Graphics

Computer graphics is concerned with object geometry and appearance models, scene description, and image generation. As the field of computer graphics and virtual environments has evolved, there

has been increasingly interest in acquiring and tracking the motion of users, as well as in simulating dynamic and changing environments. We categorize the research challenges concerning motion in computer graphics into two major areas: interactive display and motion generation.

Interactive display. In traditional polygon rendering of massive geometry datasets, several rendering acceleration techniques have been proposed, including visibility culling, model simplification, and image-based techniques. Over the last few years, model simplification for static environments has been extensively studied; see [20, 26] and references therein. However, little has been done for geometric simplification of complex, dynamic environments. Erickson and Manocha [17] have recently proposed an automatic technique based on ''hierarchical levels of detail'' (HLOD) to automatically simplify large dynamic environments. It works well for datasets with modest amount of movement. However, further study is required for view dependent simplification of dynamic environments consisting of relatively large movement. Similarly, little research has been done on visibility or occlusion culling in dynamic environments. Although kinetic data structures (as discussed in Section 3.1) for spatial decompositions such as binary space partition trees have been proposed [4, 2] and implemented, several robustness and efficiency issues must be addressed before they become suitable for general use.

A difficulty in adapting the known simplification and occlusion-culling techniques to handle dynamic environments is that they require sophisticated preprocessing steps, which makes the updates quite expensive. Simplification and occlusion-culling techniques that involve little preprocessing are needed for dynamic environments. Furthermore, often a data structure is specially designed for each rendering acceleration technique. It is impractical to integrate all different types of rendering acceleration techniques using multiple data structure of real-time rendering of complex geometric datasets. Thus it remains a challenge to design a unified data structure for combining multiple rendering acceleration techniques (e.g. visibility culling, geometry simplification and image-based rendering) for interactive display of a massive environment, using only one type of data structure to achieve near-optimal performance.

Besides polygon-based rendering, volume rendering and image based rendering have recently received much attention. How to extend these techniques to dynamic environments remains a challenging problem.

Extending the frontier of man-machine interaction, haptic rendering augments graphical and aural display by enhancing the level of immersion and accommodating the sense of touch. In order to create a sense of touch between the user's hand and a virtual object, contact or restoring forces are generated to prevent penetration into the virtual model. This is computed by first detecting if a collision or penetration has occurred, then determining the (projected) contact point on the model surface. If penetration has occurred, then it is often necessary to compute the penetration depth for computing the restoring forces. Some of open research issues are (1) design of proximity query algorithms capable of running at the KHz rates required by commercial haptic devices, (2) haptic display of textured surfaces — i.e. fast and accurate simulation of friction; (3) simulation of deformation between flexible bodies for enabling real-time haptic interaction.

In order to immerse a user in a virtual environment (VE), motion tracking and prediction is essential for man-in-the-loop interaction with VEs. Tracking user motion, including head, hand and

body movement, for real-time mobile interaction with virtual environments is of great importance. Augmented reality (superimposing a virtual scene over a real, physical environment), registering motion of synthetic environment with the physical world undergoing motion presents another challenge to researchers in motion modeling.

Motion generation. The following four mechanisms are commonly used to generate and synthesize motion for computer graphics and virtual environments: animation by keyframing and kinematics, playback using motion capture, physically-based simulation, and high-level behavior control.

A scalable and automatic technique for motion editing is one of the active research areas for animation using keyframing. Other problems include specifying and representing deformation for kinematics-based approach to synthesize motion. Despite many years of investigation in this area, developing accurate and robust methods for stable motion tracking and registration based on captured data remains an open research challenge. Once the captured data is available, fitting the recorded motion with any animated character also presents several interesting problems.

For physically-based simulation, there are several outstanding research issues. Some of them include dynamic simulation and control of complex systems (articulated, flexible bodies), modeling of heterogeneous and deformable materials, use of simulation level of details (as an analog to geometric levels of detail), and the design of proximity query algorithms based on multiresolution representations. Several of these issues have already been discussed in Section 3.3. Some of research problems for high-level behavior control include task and motion planning of autonomous agents, control of articulated bodies, and high-level behavior learning based on captured motion libraries.

3.6 Computer Vision

Shape and motion. Vision processes data representing motion in the physical world. Sensor noise, calibration, and uncertainty are novel issues that have to be addressed. Motion analysis in computer vision can very crudely be divided into scene-based and object-based approaches.

Scene-based analysis based on generic, model-free methods has been studied from the 1980s onwards and has led to successful systems for camera pose estimation and image stabilization. This method usually exploits either rigid-body scene assumptions or optic flow to estimate gross motion characteristics. A common approach in this area is to extract structure from motion, assuming a single moving camera in a static scene [25, 19]. While challenges remain in these areas, commercial codes are starting to become available to solve such problems (e.g., `http://www.2d3.com/`).

Most of the deep motion-related research problems in vision arise when the scene must be interpreted in terms of objects undergoing independent motion. Consequently, doing motion analysis successfully for computer vision often relies on having sufficiently expressive shape (or, more generally, appearance) models to describe the objects being viewed. Certain classes of shapes have been successfully modeled. These include rigid, near planar shapes, and those whose contours or intensities can be modeled using linear combinations of principal components. There have also been successes with 3-D jointed structures as well as 2.5-D 'cardboard cutout' representations. However, there remain large classes of objects which we do not yet know how to model well, including such pragmatically important ones like people wearing loose clothing. Much of the difficulty in object

tracking comes as much from the challenge of devising a suitable appearance model as from the motion analysis itself.

In the past most of the shape models used have been hand-crafted, parameterized models [16] and the methods developed for motion analysis have been deterministic in nature (e.g., physics-based methods). Recently, to overcome the obvious limitations of hand-crafted models, to generalize the representation of motion in terms of parameters that follow distributions, and to improve the coupling between motion estimation and motion recognition, statistical methods have become popular [21, 11]. Such methods promise new capabilities for segmentation, parameterized shape representation, motion estimation and motion recognition.

Within this new class of statistical methods the following open research problems need to be addressed.

Segmentation, grouping and initialization. While there has been a significant improvement in recent years in our ability to track objects with complex appearance and motion models, the problem of initialization — the matching of an object model to this first frame of a sequence as a precursor to tracking — has lagged behind. Initialization is typically treated as a separate algorithm distinct from the subsequent tracking, and often relies on generic segmentation techniques. Furthermore, initialization is often attempted using a single image whereas it might be much easier to do given several frames; it is clear that we need a better understanding of how to merge the problems of initialization and tracking to arrive at a continuum between them. Initialization is also clearly related to grouping methods; for example initializing a jointed-limb model can be viewed as the task of grouping the individually moving limbs into a coherent whole. We have as yet no good algorithms for incorporating this grouping into the initialization/tracking process, perhaps in order to select the form of a shape model appropriate to an unknown object. Finally, the problem of coupling shape models with statistical methods in an effort to automate the estimation of the priors for improved segmentation and model initialization has only recently begun to be studied [13].

Statistics and learning. In common with other fields such as speech recognition, the computer vision community has increasingly come to understand that methods which take advantage of learned statistical models are more robust and powerful than those which rely on heuristics. Although there are learning based methods for estimating low-level motion parameters, basic research needs to be done on coupling the low level motion parameters estimated by the above methods with the semantic interpretation of these parameters. An example of such needed integration is the problem of Gesture and Sign Language Recognition [31].

While motion analysis has so far been studied under the assumption of fairly well lit objects that follow the optical flow assumption, very little research has been done on motion analysis under varying lighting conditions. This is clearly a very important problem that is essential to allow robust motion tracking.

Multiple scales. Objects in the world are often naturally represented at multiple scales. This is true of object shape, which may lend itself to hierarchical description; object texture: for example a piece of paper may be well represented as having uniform texture from a distance and as containing

discrete letters close up; and object motion, which is typically well approximated by a simple linear process at short timescales, while these linear motions compose over longer timescales to form complex behaviors which may be best described using grammars over discrete spaces. We have some algorithms for dealing with multiple scales but no good overall theory, and we do not yet fully understand the conditional dependencies which relate the different scales. The study of principled statistical algorithms to deal correctly with inference of motion over multiple scales is also in its infancy.

Summary. In some ways, motion analysis is one of the success stories of recent computer vision research. Around ten years ago memory sizes and computing speeds became adequate to deal with lengthy image sequences and since then tracking has made great strides: simple 'blob' trackers which follow the gross motion of non-overlapping objects against relatively static backgrounds are now a standard component of robust systems which can run unattended for months. As soon as the complexity of the problem is increased in any dimension, however, current solutions break down, so there is plenty of research left to do, from following the motion of complex non-rigid objects to interpreting and summarizing extended behaviors, to coupling vision-based motion trackers with other modalities, such as speech, for improved activity recognitions.

3.7 Robotics

As in the previous section, robotics must deal with motion in the ''real'' world — and not just with idealized models of such motion in a virtual world. As a consequence, besides the usual criteria such as efficiency and simplicity, the performance of motion algorithms in robotics must be judged based on how accurately they model the real world (for a class of tasks) and how robust they are to the uncertainties.

Modeling motion. Some of the key difficulties of modeling motion in robotics include model selection and parameter estimation. Despite extensive work in mechanics over the last few centuries, there are numerous every-day objects that robots need to reason about and manipulate that are not easy to model using known laws. The fundamental problem lies at the fact that the parameters needed for the models are usually unknown and difficult to acquire. For instance, our ability to model the motion of an object rolling on the ground is limited by the difficulty of knowing the surface friction at the interface and even the inertial properties. Or, consider the problem of modeling a deformable object, say a human organ like the liver, in surgical robotics. How is the deformation of the liver to be modeled? In order to tackle these problems, one has to address the following issues:

- Algorithms for estimating parameters of motion models from observations. These can often lead to ill-posed inverse problems, requiring sophisticated numerical techniques.
- Techniques for representing, propagating, and reasoning about uncertainty in the motion. Kalman and particle filters are a couple of examples of representing and propagating the uncertainties, but we also need algorithms for incorporating these uncertainties in collision detection and impact problems.

- New empirical models. For many complex problems, empirical models are the only option to model motion. For example, in order to model the deformation of nonuniform elastic objects, rather than reconstructing the internal structure of the object so that known models of elasticity can be applied, it may be preferable to construct an empirical model of the deformation behavior of the object. Essentially all friction and impact laws used in practice are empirical, and they are applicable only for a small class of tasks.

Many of the difficulties in modeling motion are because of the the *constraints* that the motion must satisfy. The physical laws governing the motion are obvious constraints. These laws are typically continuous and are modeled using differential equations. However, the motion algorithms discretize them. Despite the recent progress in discretization of differential equations, there many issues remain unsolved, including the ability to extend them to deformable shapes. Whether the kinetic data structures, discussed in Section 3.1, can be used to discretize differential equations remains an interesting open question. Contact constraints are another tpe of constraints that are hard to model. For instance, simulating the roll-slide motion during smooth contact is a challenge to both collision detection algorithms (especially those relying on polyhedral approximations and bounding volumes) and to traditional simulation algorithms (especially those relying on constraint stabilization). How to represent local deformations at contact is another challenging problem — localized (''point'') contacts produce very large stresses, which locally deform apparently rigid objects and can produce important elastic effects such as stick-slip oscillations.

Collision detection. A central problem in robotics is detecting collision between many moving objects. It has been extensively studied both in robotics and computational geometry. The work in computational geometry has focused on theoretical approaches, which are at times complex to implement, while the work in robotics has lacked a general approach that works well in all cases. Commonly used in practice are methods that track the closest pair of features between two moving objects and update the closest pair by a local computation after each time step; an underlying assumption is that, if the time step is chosen sufficiently small, then the closest pair can be updated incrementally and efficiently. Feature trackers work well for simple convex objects, or combinations of a small number of such. Hierarchical representations of objects have been incorporated to this approach to handle more complex objects, and kinetic data structures (Section 3.1) have been used to alleviate some of the problems of the fixed-time-sampling approach. Alternatively, hierarchies of simple bounding volumes (spheres or boxes) have been used to enclose shapes; these hierarchies are refined only to the coarsest level necessary to establish that the two objects do not intersect.

The above approaches, however, are not suitable for detecting collision between many moving objects, especially if each of them is moving with different speed, because one has to perform collision testing for every pair of objects. Instead a global approach is needed. Recently a few global approaches are proposed for detecting collision between many moving polygons in the plane, which maintain a tiling of the common exterior of the polygons into flexible cells, so that the polygons are known to be disjoint until one of the cells self-collides. They maintain additional information that makes it easy to predict when a cell self-collides, and to update the tiling when that occurs. Although this approach looks promising in the plane, there are many stumbling blocks in extending

it to 3-space. Even less is known on detecting collision between deformable objects or for detecting self-intersection in a deformable object.

A different problem is decide what to do when a collision is detected. For example, in virtual-reality applications such as computer games, digital battlefields, haptics interfaces, etc., it is critical that reaction to collision detection is close to reality — Should an object penetrate the other? Should one of the objects bounce back? Should the objects deform or break? This raises several interesting questions that combine geometry with physical models of the objects involved.

Planning motion. A typical motion-planning problem asks for computing a collision-free motion between two given placements of a given robot in given environment. In its simplest form, one assumes that the environment is fully known and there are no constraints on the motion of the robot except that it cannot collide an obstacle. The problem is typically solved in the *configuration space*, in which each placement (or configuration) of the robot is mapped as a point; see [24]. The *free configuration space* F is the subset of the configuration space at which the robot does not intersect any obstacle. The robot can move from an initial configuration to a final configuration without intersecting an obstacle if these two configurations lie in the same connected component of F. Planning a collision-free path thus reduces to connectivity and other topological questions in F. Numerous general techniques and techniques for low-dimensional configuration spaces have been developed [8, 18]. This has also led to several interesting topological questions on configuration spaces, which are typically represented as semi-algebraic sets (finite Boolean combinations of solution sets of polynomial equalities and inequalities).

The dimension of configuration space depends on the degrees of freedom of the robot, and it can be quite high. Computing high-dimensional configuration spaces exactly is impractical [8]. Therefore techniques to compute an approximate representation of F are needed. Much of the difficulty in approximating F is in understanding the topology and in simplifying the topology of F. Recently, Monte Carlo algorithms have been developed for representing F by 1-dimensional networks, called *probabilistic road maps* (PRMs) [22, 23]. Intuitively, this network is an approximate representation of the *road map*, a 1-dimensional network that captures the connectivity information of F. These methods sample points in F and connect them by an edge if they can be connected by a simple path inside F. Despite several heuristics to sample well the points in F, better sampling strategies are needed to handle narrow corridors and other difficult areas in F, so that the connectivity of F is preserved. Variants of such methods have already been described in Section 3.4. PRMs can quickly find a simple collision-free path between initial and final configurations in many situations. But there is no absolute guarantee that they will do so – in fact, very few general purpose techniques are known for establishing that paths do not exist.

In the above discussion, we assumed that there were no constraints on the motion of the robots, which in practice is not the case. Various nonholonomic constraints such as bound on the maximum velocity or acceleration may exist on the moving robot. Planning a collision-free in presence of these constraints is considerable harder, and relatively little is known except various *ad-hoc* approaches. Better techniques that are practical as well as theoretically sound are need for nonholonomic motion planning.

In some applications even more challenging problems arise. If the obstacles are also moving, the

free configuration space has to be updated dynamically. Also flexible objects such as elastic bands, rope, or cloth cannot be properly represented with a finite number of degrees of freedom. How can one represent configuration spaces of such objects is big challenge.

3.8 Spatio-Temporal Databases

Efficient indexing schemes that support various queries are central to any large database system. Most existing database systems assume that the data is constant unless it is explicitly modified. For example, if the value of the age field is set to 40 years, then this age is assumed to hold (i.e., 40 years is returned in response to a query) until explicitly updated. Such systems are not suitable for representing, storing, and querying continuously moving or varying objects; either the database has to be continuously updated or a query output will be obsolete. Motivated by applications in digital battlefields, air-traffic control, and mobile communication systems, methods for indexing dynamic attributes are needed, which guarantee efficient access to moving objects. A basic scenario is one in which the database system receives trajectories for a large population of objects capable of continuous movement; new objects may arrive, existing objects may leave, and that the positions of existing objects may be updated dynamically. The objects, which can assume to have rigid shapes, may be represented as points. People with mobile phones, people in Internet-worked cars, airplanes, etc. The movement of the objects may be constrained by stationary or moving objects, the movement of the objects may be confined to networks (e.g., cars confined to a road network), and the motion of objects may be highly correlated. The goal is to answer various queries based on the locations, trajectories, and topology, and to explore patterns in motion of objects. Since the data is quite large and resides on disk, one cannot scan the whole data to answer a query.

Instead of continuously updating the position of a moving object, a better approach is to represent the position as a function $f(t)$ of time, so that changes in object position do not require any explicit change in the database system. With this representation, the database needs to be updated only when the function $f(t)$ changes. Recently there has been some work on extending the capabilities of existing database systems to handle moving-object databases (MOD); see, for example, [27, 28, 15].

Methods such as KDS, developed for simulating some structures/phenomena over time, are not always suitable for answering queries on the database of moving objects. The queries might relate either to the current configuration of objects or to a configuration in the future — in the latter case, we are asking to predict the behavior based on the current information. In either case, tracking the current configuration of the objects would be unnecessary and expensive. Another approach to handle motion is to regard the time as another spatial axis and reduce the problem to a static problem in one higher dimension. For example, a set of points moving in the plane can be regarded as lines in $\mathbb{R}^3$. This approach has the advantage that the data structure need not be updated unless the trajectory of an object changes. However, it suffers from a number of problems. First, the approach expects the trajectory of the points to be known in advance. Second, it reduces the problem to one higher dimension, which makes the algorithms slower.

Here are a few different types of queries that one may want to answer on moving objects.

Location based queries. These queries include range and proximity queries. For example, given a rectangle R and a time t, report all objects that will be inside R at time t; given a rectangle R and a time interval $[t_1, t_2]$, report all objects that will pass through R in the time interval $[t_1, t_2]$; given a point p and a time interval $[t_1, t_2]$, report the objects that would be nearest to p in that time interval. Despite a flurry of papers on indexing moving objects for answering location-based queries, the problem remains largely open. The theoretical results, based on partition trees and KDSs, are too complex to be practical. The practical indexing schemes are based on quad-trees, R-trees, and their variants. They do not always work well, and their performance deteriorates with time and the data size. In order to improve the performance of an indexing scheme, Some of them have introduced the notion of *horizon* that gives the duration for which the current index performs well. They suggest to rebuild the index periodically or to adjust it when the trajectories are updated (see e.g. [29]). Although these approaches provide better query performance, it is not always practical to rebuild the index periodically.

Continuous queries. In the above queries, the query range was spatially fixed. One could also ask queries when the location of the query range is also moving. For example, keep track of all cars within 5 miles. Of course, such a query can be answered by asking location-based queries repeatedly, but it would be very expensive. One would like to combine KDS with procedures for location-based approaches to answer such queries. The known lower bounds on range-searching data structures imply that it is hard to answer such queries efficiently in the worst case. But can one exploit the constraints on motion to answer them efficiently?

Trajectory-based queries. These queries involve the topology of trajectories and derived information such as velocity of the objects. These queries are deemed critical, but also rather expensive. Here is a typical query: Which of two objects will be within one mile during the next ten minutes? Which of the objects are heading *East*, or which of the objects are moving at speed greater than 65 miles/hour. An even harder query is: report all objects whose speed doubles in the next ten minutes.

Dynamization. It is an inherent characteristic that new objects arrive and that the positions of existing objects are updated dynamically. Hence, the index should support fast insert, delete operations. Although basic spatial indexing schemes, such as R-trees or kd-trees, support insertion/deletion of objects, the problem is considerably harder for indices handling moving objects because they have to store additional information to predict the motion of objects. This information is expensive to update as objects are inserted or deleted.

The current indexing schemes basically delete and re-insert a point whenever the trajectory of an object changes. In may applications, the trajectory of an object is known in advance. Can this information be used to avoid explicit deletion and re-insertion of the object.

Uncertainty. Since in many applications the position of objects is being received through a sensor such as GPS, uncertainty in position and velocity of the objects is inevitable. Indexing schemes that can handle uncertainty are needed. A simple model of uncertainty is a to allow an interval of values

for each of the parameters defining the dynamic attributes of an object. In this model, the uncertainty in position corresponds to a 'ball' in which the object can lie, and the uncertainty in velocity corresponds to a 'cone' in which the object lies. These simple models have been incorporated in a few indexing schemes, but a more systematic study is needed. More sophisticated parametric models have been developed for modeling uncertainty. Although these methods have been successfully used for tracking moving objects, they have not been used in the context of databases. How to incorporate them in indexing schemes is a challenging open problem.

3.9 Mobile Wireless Networks

A world of 'anytime, anywhere, anyhow' computing is being created by a unique combination of two powerful technological trends: rapid component miniaturization and the emergence of high speed wireless communication. The recent boom in the use of cell phones is merely the beginning of a much larger trend towards broadband wireless networks that carry not only voice, but all kinds of multimedia data. Wireless data networking presents a multitude of challenges that are distinct from their fixed infrastructure networking counterpart:

- [*Heterogeneity.*] There are a variety of wireless technologies (e.g. modulation schemes and spectrum bands), diverse devices (e.g. laptops, PDAs, data enabled cell phones and pagers), communication protocols (e.g. several 3G standards for wireless and SMS). Finally the channel conditions under which wireless communication occurs vary widely.

- [*Resource Constraints.*] The nature of wireless technology and devices presents stringent resource constraints, such as the available spectrum (number of channels, codes, slots, etc.) and the power (total and received power), among others.

- [*Mobility.*] But the most distinguishing aspect of a wireless network is the mobility of its clients. Dealing with mobility presents new problems (handoffs, paging clients, etc.) as well as it complicates existing ones (such as power allocation schemes where channel conditions vary with mobility patterns). In fact, any structuring mechanism provided by the network that exploits the local topology of the nodes has to be adapted to motion.

Mobility and routing. A *mobile ad hoc network* (MANET) is an autonomous system of mobile routers and hosts connected by wireless links. The network is self-organizing and self-configuring, with nodes establishing the necessary routes among themselves without the help of a fixed infrastructure. Since the communicating hosts may be some distance apart, multiple network hops through intermediate nodes may be required to complete the route. The hosts themselves act like 'mobile routers,' and cooperatively forward messages not addressed to them. As these hosts move around, the routing protocol of the network must adapt its routing decisions to maintain communication. The rate of topology change may be quite dramatic in some ad hoc networks.

Since the topology of the network is constantly changing, the routing protocols must cope with frequent link failure and disconnections. The distribution of up-to-date information can easily overload the network, while out-of-date information can drive a network into instability. Thus, the

shortest path routing used commonly in fixed infrastructure networks may be the wrong routing algorithm for ad hoc networks. Instead, discovering and routing on 'long lived' routes may be more desirable.

In large-scale ad hoc networks, it may be infeasible, and even undesirable, for each host to maintain up-to-date information about the locations of all hosts and the topology of the entire network. Perhaps a more useful approach will be a statistical modeling of the movement of individual hosts. A grand challenge in the field, which may be currently just academic, but which will have tremendous impact on systems and performance, is the following: *Can we model the mobility patterns of users at varying levels of granularity*? The question should be explored in the context of modeling arrival patterns of customers to service queues, web server request sizes, etc. It could be refined in many ways — e.g. does an individual users' mobility pattern fit a small number of short random walks on the plane? Can the number of roaming users in any cell at any moment be predicted without understanding individual users' mobility pattern? This grand challenge may involve input from geometers on finding a small set of attributes of the users' trajectories on the globe that are most predictive of the users' movements. One of the aspects of this grand challenge that is interesting is its 'internet' scale of users and resource usages.

Mobility and network layers. Another approach to study the impact of mobility on networking infrastructure is to study its effect on the various architectural layers of the network. We list below for each layer some fundamental issues that are worth exploring.

1. [*Physical Layer.*] How does one perform resource scheduling at the physical layer in order to optimize performance for a given mobility pattern. For instance, how does one optimize across issues of power consumption, error correction, coding, rate control, channel assignment, and so on.

2. [*MAC Layer.*] How does one discover resources in the presence of *ad-hoc* mobility, such as reachability testing, topology determination, and so on.

3. [*Transport Layer.*] How and to what extent must one implement end-to-end TCP properties in presence of host mobility? How does high vs. low mobility affect persistence of TCP connections and fragmentation? Packet loss probabilities are significantly higher in wireless networks than in the wired networks due to widely varying channel conditions that are compounded by users' mobility.

   Specifically, the TCP reacts quite violently to packet losses, with its exponential backoff mechanism, because it interprets the loss of packet to mean congestion in the network. However, in the wireless networks, the packet losses are just as likely the result of transmission loss as congestion, so how should the TCP rate control mechanism be modified?

4. [*Network Layer.*] How should one balance the cost of 'redirection' vs. 'paging,' in networks for accessing a user by better understanding of mobility models?

5. [*Application Layer.*] How does one support location specific queries on databases of yellow pages or services? How does one support a geographically connected community of users

or ‘buddies’? These involve testing geographic neighborhoods based on distance constraints where distance updates are obtained by pinging.

Many additional issues are likely to arise as wireless data networking and web evolve to become more widespread. While many of the issues listed above are algorithmic, they are best explored by collaboration with electrical engineers and computer scientists who work on different layers of wireless networking in order to be effective. Such collaborations represent an opportunity for the theoretical computer science community to formulate and solve the fundamental problems of interest in mobility and related issues in wireless networking. Finally, there is clearly a large overlap between the problems discussed here and those in Section 3.8 on spatio-temporal queries for mobile data.

4 Challenges

In this section we review a number of challenging issues that cut across many areas.

Motion representation and analysis. This covers most of the topics we have already presented in earlier sections:

Uncertainty. Whenever we try to model motion in the physical world we must deal with uncertainty. So far only the computer vision and robotics communities have developed refined tools for the probabilistic treatment of object motion. Such ideas could also prove very very useful in the temporal database and networking domains.

Robustness. Many geometric algorithms suffer from robustness problems because of mismatch between fast floating-point computer arithmetic, which is finite precision, and the semantics of real analysis. Algorithms fail when an approximate computation yields an incorrect or even inconsistent qualitative property (often a topological property). In the context of kinetic algorithms, uncertainty in the input makes the problem even more acute.

Aggregation. Related to uncertainty is the issue of statistical models for representing aggregate motion, say the motion of cars on the freeway, or a flock of birds. We may be able to reason reliably about the high-level behavior of the system, even though we have large uncertainty about the motion of each individual object.

Simplification. While shape simplification methods are highly developed, much less is known about motion simplification. A molecular dynamics simulation, for example, includes all the atomic vibrations caused by thermal noise, which can obscure the large-scale structure of the motion that we want to understand.

Hierarchical representations of complex motion. Related to simplification is the issue of devising hierarchical and multi-resolution motion representations.

Marrying the continuous and the discrete. Most physical systems evolve following continuous evolution laws, yet their evolution is punctuated by discrete events, such as collisions, that can alter these laws. Historically the communities that have studied the continuous and discrete aspects of this problem have been distinct (scientific computing vs. discrete algorithms). A much tighter integration between the two can lead to substantial progress.

Developing efficient algorithms. Clearly a lot remains to be accomplished on developing efficient motion algorithms and the tools for their analysis.

Trade-off between realism and efficiency. Little has been done so far to formally explore trade-offs between accuracy and efficiency. Such analyses can benefit all of the areas we have been discussing.

Decentralization. Distributed motion algorithms are equally important for large-scale simulations and for low-power mobile devices in situations such as ad hoc networking or sensor nets.

Querying motion. Many fundamental questions remain on how best to sense and organize mobile data when the goal is to answer certain queries about the mobile system (and there is no need to know the full state of the system in between).

Motion Integration. Though each of the areas we discussed has good reasons for the motion representations it has chosen, this diversity hinders the development of end-to-end systems where techniques form several disciplines need to process motion data in an integrated fashion. Currently big gaps exist between the motion representations best suited for visual motion analysis and those for motion generation, for example. We feel it is important to define a number of fundamental motion representations that can be used across areas and which can form the basis for implementing integrated applications.

5 Recommendations

- Funding opportunities are needed to encourage further work in these areas, either as a separate initiative or as continued funding from the relevant areas within NSF and DOD.

- To facilitate cross-disciplinary research, we recommend a solicitation on the theme of an integrated, end-to-end system dealing with motion. The focus of the research will be the collaboration of motion researchers from different communities in a system that exploits motion representations and algorithms across several fields. Example might be: real-time robotic mimesis based on motion capture (a robotic actor imitating the motions of a real actor in real-time), real-time monitoring of ad hoc mobile or cellular network performance, multiple vehicle tracking using a distributed sensor net, the simulation of deformable objects in large-area contact, multi-resolution molecular dynamics, etc.

- It seems premature to establish a journal or annual conference series in this area, but at the very least there should be workshops on modeling motion at regular time intervals, say, once every other year. The last workshop was an invitation-only, direction-finding session; what is needed now is a forum for collecting new work in the area, unifying the work being done in different communities, and fostering interdisciplinary collaboration.

  Agarwal and Guibas are planning to organize a week-long workshop in Fall 2002 at DIMACS, Rutgers University, New Jersey, as part of the special year on computational geometry. It will be an open workshop in which participants will present recent work on motion modeling. There will also be plenty of time for participants to discuss open problems and to work together.

[1] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tiling for kinetic collision detection. In *Proc. 4th Workshop Algorithmic Found. Robot.*, 2000. To appear.

[2] P. K. Agarwal, J. Erickson, and L. J. Guibas. Kinetic binary space partitions for intersecting segments and disjoint triangles. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 107–116, 1998.

[3] P. K. Agarwal, L. J. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving point set. In *Proc. 5th Workshop Algorithms Data Struct.*, volume 1272 of *Lecture Notes Comput. Sci.*, pages 31–44. Springer-Verlag, 1997.

[4] P. K. Agarwal, L. J. Guibas, T. M. Murali, and J. S. Vitter. Cylindrical static and kinetic binary space partitions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 39–48, 1997.

[5] P. K. Agarwal and S. Har-Peled. Maintaining approximate extent measures of moving points. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, 2001.

[6] P. K. Agarwal and M. Sharir. Arrangements and their applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[7] M. J. Atallah. Some dynamic computational geometry problems. *Comput. Math. Appl.*, 11(12):1171–1181, 1985.

[8] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Internat. J. Robot. Res.*, 10:628–649, 1991.

[9] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.

[10] M. Bern and P. Plassmann. Mesh generation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 291–332. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[11] A. Blake, B. North, and M. Isard. Learning multi-class dynamics. In *Advances in Neural Information Processing Systems*, volume 11, pages 389–395. MIT Press, 1999.

[12] R. Bryant, H. Edelsbrunner, P. Koehl, and M. Levitt. Inclusion-exclusion formulas for the area derivative of a space-filling diagram. manuscript.

[13] T. Chen and D. Metaxas. Image segmentation based on the integration of markov random fields and deformable models. In *Proceedings of MICAI 2000*, pages 11–14, 2000.

[14] H.-L. Cheng, T. K. Dey, H. Edelsbrunner, and J. Sullivan. Dynamic skin triangulation. *Discrete Comput. Geom.*, 25:525–568, 2001.

[15] J. Chomicki and P. Z. Revesz. A geometric framework for specifying spatiotemporal objects. In *Proc. 6th Intl. Workshop on Time Representation and Reasoning*, pages 41–46, 1999.

[16] D. DeCarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 32:99–127, 2000.

[17] C. Erickson and D. Manocha. Simplification culling of static and dynamic scene graphs. Technical Report TR98-009, U. North Carolina, Computer Science Department, Chapel Hill, NC, 1998.

[18] D. Halperin, L. E. Kavraki, and J.-C. Latombe. Robotics. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 755–778. CRC Press LLC, Boca Raton, FL, 1997.

[19] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[20] H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.

[21] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[22] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 353–362, 1995.

[23] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. Robot. Autom.*, 12:566–580, 1996.

[24] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[25] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[26] J. Popovic and H. Hoppe. Progressive simplicial complexes. *Computer Graphics*, 31(Annual Conference Series):217–224, 1997.

[27] A. P. Sistla and O. Wolfson. Temporal conditions and integrity constraints in active database systems. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 269–280, 1995.

[28] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proc. Intl Conf. Data Engineering*, pages 422–432, 1997.

[29] J. Tayeb, O. Ulusoy, and O. Wolfson. A quadtree-based dynamic attribute indexing method. *The Computer Journal*, pages 185–200, 1998.

[30] A. Ungör and A. Sheffer. Tent-pitcher: a meshing algorithm for space-time discontinuous Galerkin methods. In *Proc. 9-th Int. Meshing Roundtable*, pages 111–122, 2000.

[31] C. Vogler and D. Metaxas. A linguistics-based approach to american sign language recognition. In *International Conference on Gestures: Meaning and Use*, Porto, Portugal, 2000.